

# Introduction to SciNet

SciNet HPC Consortium  
Compute Canada

May 14, 2014

**Don't  
Panic**

# Part I

## ABOUT SCINET

# SciNet is ...

... a consortium for High-Performance Computing consisting of researchers at U. of T. and its associated hospitals.



There are 6 consortia in Canada that provide HPC resources to Canadian academic researchers and their collaborators.



# SciNet is ...

*... where to take courses on computational topics, e.g.*

- Parallel programming courses
- Scientific Programming with Modern FORTRAN
- GPGPU with CUDA
- Parallel I/O
- Intro to SciNet
- Linux Shell
- Research Computing with Python
- Scientific Computing Course (credit for physics/chemistry/astro grads)
- Ontario HPC Summerschool

<https://support.scinethpc.ca/education>

# SciNet is ...

... where to meet other users at the SciNet User Group meetings.

- ▶ SNUGs every 2nd Wednesday of the Month.
- ▶ 1 or more TechTalks
- ▶ Free pizza!
- ▶ <https://support.scinet.utoronto.ca/education>

... where you go for SciNet Developer Seminars

# SciNet people

## Analysts

- Bertrand Brelier
- Jonathan Dursi
- Daniel Gruner (CTO)
- Scott Northrup
- Erik Spence
- Ramses van Zon

## Hardware and Systems

- Joseph Chen
- Jason Chong
- Leslie Groer
- Neil Knecht
- Chris Loken (CTO)
- Jaime Pinto
- Ching-Hsing Yu

## And

- Technical director  
Prof. Richard Peltier
- Business manager  
Teresa Henriques
- Project coordinator  
Jillian Dempsey

## Part II

### SCINET SYSTEMS

# Compute Resources at SciNet

## General Purpose Cluster (GPC)



[http://wiki.scinethpc.ca/wiki/index.php/GPC\\_Quickstart](http://wiki.scinethpc.ca/wiki/index.php/GPC_Quickstart)



# Compute Resources at SciNet

## General Purpose Cluster (GPC)

- 3864 nodes with 8 Intel x86-64 cores @ 2.53Hz
- 30,912 cores
- 328 TFlops
- 16 GB RAM per node ( $\sim 14$ GB for user jobs)
- 16 threads per node
- Operating system: CentOS 6
- Interconnect: InfiniBand
- #16 on the June 2009 *TOP500* (Now at #148)
- Third ranked Canadian system.

[http://wiki.scinethpc.ca/wiki/index.php/GPC\\_Quickstart](http://wiki.scinethpc.ca/wiki/index.php/GPC_Quickstart)

# Other Compute Resources at SciNet

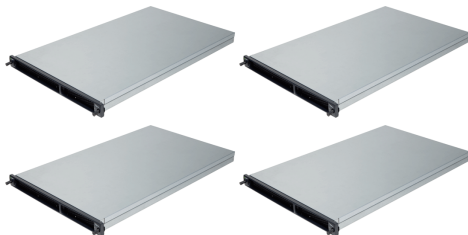
## Tightly Coupled System (TCS)



## Power 7 Linux Cluster (P7)



## GPU Devel Nodes (ARC)



## Blue Gene/Q



# Storage Resources at SciNet



## Disk space

- 1.4 PB of storage in 1790 drives
- Two controllers each delivering 4-5 GB/s (r/w)
- *Shared* file system GPFS on all systems
- Your files go in /home/g/group/user (\$HOME) and /scratch/g/group/user (\$SCRATCH).

## Storage space

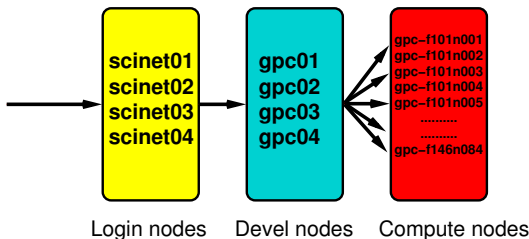
- HPSS: Tape-backed storage expansion solution.  
1-2TB per group if they ask (*more by allocation*)

<http://wiki.scinethpc.ca/wiki/index.php/HPSS>

# Part III

## USING SCINET

# Using SciNet



## 1. Access systems: [login.scinet.utoronto.ca](http://login.scinet.utoronto.ca)

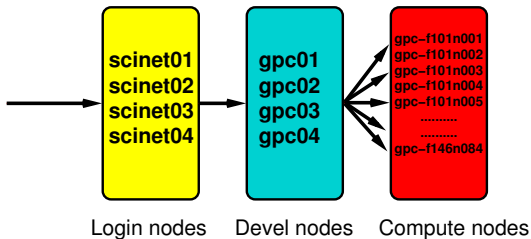
First ssh to login (not part of clusters):

```
ssh <username>@login.scinet.utoronto.ca [-Y]
```

The login nodes are gateways:

- only to be used for small data transfer
- and to proceed logging into one of the devel nodes.
- -Y specifies that you want your graphics display forwarded.

# Using SciNet



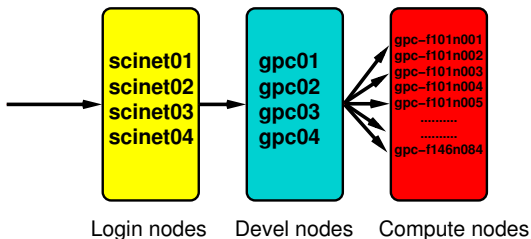
## 2. Go to the right cluster: ssh to the devel nodes

Pick a devel node. On the GPC that means: gpc01, gpc02, gpc03, gpc04. These are aliases for longer node names. E.g.

```
ssh gpc03 -Y
```

gets you to the gpc development node named gpc-f103n084.  
The command `gpc` picks a node for you.

# Using SciNet



## 3. Submit a job to the compute nodes

*E.g.*

```
cd $SCRATCH/jobdir  
qsub jobscript.sh
```

Wait, didn't we skip some steps?

# Software and Libraries (1)

Once you log into devel nodes, what software is already installed?

- Other than essentials, all software installed using module commands.
- sets environment variables (LD\_LIBRARY\_PATH, PATH, ...)
- Allows multiple, conflicting versions of package to be available.
- More on *Software and Libraries* page of wiki.

[http://wiki.scinethpc.ca/wiki/index.php/Software\\_and\\_Libraries](http://wiki.scinethpc.ca/wiki/index.php/Software_and_Libraries)



# Software and Libraries (1)

Once you log into devel nodes, what software is already installed?

- Other than essentials, all software installed using module commands.
- sets environment variables (LD\_LIBRARY\_PATH, PATH, ...)
- Allows multiple, conflicting versions of package to be available.
- More on *Software and Libraries* page of wiki.

```
gpc-f103n084-$ module avail
```

```
-----/scinet/gpc/Modules6/Modules/versions-----  
3.2.8  3.2.9  
-----/scinet/gpc/Modules6/Modules/3.2.9/modulefiles-  
dot                modules                use.own  
module-cvs         use.deprecated  
module-info        use.experimental  
-----/scinet/gpc/Modules6/Modules/modulefiles-  
ImageMagick/6.6.7(default)  
R/2.13.1(default)  
R/2.14.1  
ROOT/5.30.03(default)  
ROOT/5.32.00  
Xlibraries/X11-64  
abyss/1.3.2  
adios/131-openmpi-gcc(default)  
amber/10.0.30  
antlr/2.7.7  
autoconf/2.68  
automake/1.11.2  
blast/2.2.23+  
...
```

[http://wiki.scinethpc.ca/wiki/index.php/Software\\_and\\_Libraries](http://wiki.scinethpc.ca/wiki/index.php/Software_and_Libraries)

# Software and Libraries (2)

<code>module load &lt;module-name&gt;</code>	use particular software
<code>module purge</code>	remove currently loaded modules
<code>module avail</code>	list available software packages
<code>module list</code>	list loaded modules
<code>module help &lt;module-name&gt;</code>	describe a module

- We previously recommended loading modules in your `.bashrc` file. We *no longer* recommend this. It can lead to very confusing behaviour under certain circumstances.
- Instead, load modules inside your job submission script.
- Short name gives default (e.g. `intel`  $\rightarrow$  `intel/12.1.3`)

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

## Example

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$
```



# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$ module load gcc python
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$ module load gcc python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) 'intel/14.0.1 intel/14.0.0 intel/13.1.1 intel/12.1.5  
intel/12.1.3 intel/12.1.2 intel/12.1'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq intel
```

```
gpc-f103n084$
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$ module load gcc python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) 'intel/14.0.1 intel/14.0.0 intel/13.1.1 intel/12.1.5  
intel/12.1.3 intel/12.1.2 intel/12.1'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq intel
```

```
gpc-f103n084$ module load gcc intel python
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$ module load gcc python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) 'intel/14.0.1 intel/14.0.0 intel/13.1.1 intel/12.1.5  
intel/12.1.3 intel/12.1.2 intel/12.1'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq intel
```

```
gpc-f103n084$ module load gcc intel python
```

```
gpc-f103n084$
```

# Software and Libraries (3)

## Dependencies

- Modules sometimes require other modules to be loaded first.

### Example

```
gpc-f103n084$ module purge
```

```
gpc-f103n084$ module load python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) '4.8.1 gcc/4.7.2 gcc/4.7.0 gcc/4.6.1 gcc/4.4.6'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq gcc
```

```
gpc-f103n084$ module load gcc python
```

```
python/2.7.2(11):ERROR:151: Module 'python/2.7.2' depends on one of  
the module(s) 'intel/14.0.1 intel/14.0.0 intel/13.1.1 intel/12.1.5  
intel/12.1.3 intel/12.1.2 intel/12.1'
```

```
python/2.7.2(11):ERROR:102: Tcl command execution failed: prereq intel
```

```
gpc-f103n084$ module load gcc intel python
```

```
gpc-f103n084$ module list
```

Currently Loaded Modulefiles:

1) gcc/4.6.1

2) intel/12.1.3

3) python/2.7.2

# Software and Libraries (4)

## Commercial Software?

- SciNet has an extremely large and broad user base ( $\sim 1000$  users)
- The only commercial software that SciNet installs is software that can benefit everyone:  
Compilers, math libraries and debuggers.
- No Matlab, Gaussian, IDL, ...
- Octave, Python are available.
- We can help you to install software for which you have a license.

# Compiling on SciNet (1): GPC

- From `login.scinet.utoronto.ca`, ssh to one of the four devel nodes.

```
ssh gpc04 [-Y]
```

- We recommend Intel compilers, which are

```
icc, icpc, ifort
```

for C, C++, and Fortran, respectively (from the module `intel`)

- Optimize code for the GPC machine using of at least

```
-O3 -xhost
```

- Add `-fopenmp` to the command line for OpenMP
- Compile MPI code with `mpif77/mpif90/mpicc/mpicxx`.
  - 1 OpenMPI, in module `openmpi` (v1.4.4 , 1.6.4)
  - 2 Intel MPI, in module `intelmpi` (v4.0.3, 4.1.0, 4.1.1)

# Compiling on SciNet (2): library modules

- Libraries are rarely installed in 'standard locations'.
- The name of the base directory for the library you need to access is

```
-I${SCINET_[shortmodulename]_BASE}
```

- To compile code that uses that a library from a module, add

```
-I${SCINET_[shortmodulename]_INC}
```

to the compile command.

- To link to the libraries, add

```
-L${SCINET_[shortmodulename]_LIB}
```

to the link command.



# Compiling on SciNet (3): library modules

## Example

```
scinet04$ ssh -Y gpc03
gpc-f103n084$ module list
No Modulefiles Currently Loaded.
gpc-f103n084$ pwd
/home/s/scinet/rzon
gpc-f103n084$ ls
main.c module.c
gpc-f103n084$ module load intel gsl
gpc-f103n084$ module list
Currently Loaded Modulefiles:
1) intel/12.1.3 2) gsl/1.13-intel
gpc-f103n084$ icc -c -O3 -xHost -o main.o main.c
gpc-f103n084$ icc -c -O3 -xHost -I$SCINET_GSL_INC -o module.o module.c
gpc-f103n084$ icc -o main module.o main.o -L$SCINET_GSL_LIB -lgsl -mkl
gpc-f103n084$ ./main
```

# Testing

## Why test?

- Computations are run by the **job scheduler**:
  - Not interactive.
  - Gets queued, i.e. does not run immediately.

So one cannot catch obvious bugs quickly.

- You need to test your job's requirements and scaling behaviour, so you know what to request from the scheduler.
- Best to start small and work your way up.

## How to test:

- Small test jobs can be run on the devel nodes. Rule of thumb: couple of minutes, taking at most about 1-2GB of memory.
- You can run the **ddt** debugger (or gdb or ddd) on the devel nodes.
- Short tests that do not fit on a development node, or for which you need a dedicated node, can be run through the **debug queue**.

# Submitting jobs

Every job must be submitted to a batch queue (scheduler).

- You submit jobs from a devel node in the form of a script
- Scheduling is by node. **You need to use all 8 cores on the node!**
- Must run from the scratch directory (**home=read-only**)
- Copy essential results out after runs have finished.

Limits:

- Group based limits:  
*possible for your colleagues to exhaust group limits*
- Talk to us first to run massively parallel jobs ( $> 2048$  cores)
- While their resources last, jobs will run at a higher priority than others for groups with an allocation.

# GPC queues

queue	min.time	max.time	max jobs	max cores
batch	15m	48h	32/1000	256/8000
debug		2h/30m	1	16/64
largemem	15m	48h	1	16

- Submit to these queues from a GPC devel node with

```
qsub [options] <script>
```

- Common options (usually in script):

-l: specifies requested nodes and time, e.g.

```
-l nodes=2:ppn=8,walltime=1:00:00
```

-q: specifies the queue, e.g.

```
-q batch
```

```
-q debug
```

```
-q largemem
```

-I specifies that you want an interactive session.

-Y specifies that you want your graphics display forwarded.

# GPC queues

- The debug queue is used for testing: highly recommended!
- The largemem queue is exceptional, in that it provides access to two nodes (only) that have 16 processors and 128GB of ram.
- There is no queue for serial jobs, so if you have serial jobs, **YOU** will have to bunch together 8 of them to use the node's full power. Use GNU Parallel<sup>†</sup>.
- Remember: submit your job from a devel node, NOT the login node!

<sup>†</sup> O. Tange (2011): GNU Parallel - The Command-Line Power Tool,  
;login: *The USENIX Magazine*, February 2011:42-47.

# GPC job script example

```
#!/bin/bash  
  
#PBS -l nodes=1:ppn=8  
  
#PBS -l walltime=1:00:00  
  
#PBS -N simple-openmp-job  
  
cd $PBS_O_WORKDIR  
  
export OMP_NUM_THREADS=8  
  
./openmp_example > output
```

```
$ qsub scriptname.pbs
```

# GPC job script example

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=1:00:00
#PBS -N simple-openmp-job

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=8

./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the #PBS are directed to the scheduler

# GPC job script example

```
#!/bin/bash  
  
#PBS -l nodes=1:ppn=8  
  
#PBS -l walltime=1:00:00  
  
#PBS -N simple-openmp-job  
  
cd $PBS_O_WORKDIR  
  
export OMP_NUM_THREADS=8  
  
./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the #PBS are directed to the scheduler
- qsub reads these



# GPC job script example

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=1:00:00
#PBS -N simple-openmp-job

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=8

./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the #PBS are directed to the scheduler
- qsub reads these
- In this case, scheduler looks for one node with 8 cores for 1 hour.

# GPC job script example

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=1:00:00
#PBS -N simple-omp-job

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=8

./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the #PBS are directed to the scheduler
- qsub reads these
- In this case, scheduler looks for one node with 8 cores for 1 hour.
- It gives this job the name simple-omp-job.

# GPC job script example

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=1:00:00
#PBS -N simple-omp-job

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=8

./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the #PBS are directed to the scheduler
- qsub reads these
- In this case, scheduler looks for one node with 8 cores for 1 hour.
- It gives this job the name simple-omp-job.
- Once the scheduler has found such a node, it runs the script there.

# GPC job script example

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=1:00:00
#PBS -N simple-openmp-job

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=8

./openmp_example > output
```

```
$ qsub scriptname.pbs
```

- Lines starting the `#PBS` are directed to the scheduler
- `qsub` reads these
- In this case, scheduler looks for one node with 8 cores for 1 hour.
- It gives this job the name `simple-openmp-job`.
- Once the scheduler has found such a node, it runs the script there.
  - ▶ Change directory to that from which the script was submitted.
  - ▶ Setting an environment variable
  - ▶ Running the `openmp_example` application.

# GPC queues

## Monitoring

Once the job is incorporated into the queue (this takes a minute),  
Command you can use:

- `showq` to show the queue;
- `showstart JOBID` to get an estimate for when it will run
- `checkjob JOBID` to get information on the job (quite verbose).
- `canceljob JOBID` to cancel the job.

# GPC queues

Jobs can be in one of three states:

- 1 **Running:** great, all is well (at least from a scheduler point-of-view).
- 2 **Idle:** waiting in the queue for resource. All is well (at least from a scheduler point-of-view).
- 3 **Blocked:** too many jobs submitted at the same time; scheduler will not consider these until some of the idle ones have been scheduled.

More serious errors (such as asking for a node with 16 cores), will lead to a rejection, and you will get an error with a more-or-less cryptic explanation. See the FAQ on the wiki for some typical cases and their explanation.

# Example 1 (GPC)

gpc-f101n084-\$

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi  
gpc-f101n084-$
```



# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
```

```
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
```

```
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$ cat > myjob.pbs
#!/bin/bash
#PBS -l nodes=8:ppn=8,walltime=1:00:00
#PBS -N JobName
cd $PBS_O_WORKDIR
module load intel openmpi
mpirun -np 64 ./mycode > out
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$ cat > myjob.pbs
#!/bin/bash
#PBS -l nodes=8:ppn=8,walltime=1:00:00
#PBS -N JobName
cd $PBS_O_WORKDIR
module load intel openmpi
mpirun -np 64 ./mycode > out
gpc-f101n084-$ qsub myjob.pbs
2961983.gpc-sched
gpc-f101n084-$
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$ cat > myjob.pbs
#!/bin/bash
#PBS -l nodes=8:ppn=8,walltime=1:00:00
#PBS -N JobName
cd $PBS_O_WORKDIR
module load intel openmpi
mpirun -np 64 ./mycode > out
gpc-f101n084-$ qsub myjob.pbs
2961983.gpc-sched
gpc-f101n084-$ qstat    (or checkjob 2961983, or showq -u $USER)
```

# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$ cat > myjob.pbs
#!/bin/bash
#PBS -l nodes=8:ppn=8,walltime=1:00:00
#PBS -N JobName
cd $PBS_O_WORKDIR
module load intel openmpi
mpirun -np 64 ./mycode > out
gpc-f101n084-$ qsub myjob.pbs
2961983.gpc-sched
gpc-f101n084-$ qstat (or checkjob 2961983, or showq -u $USER)
  Job id              Name              User Time Use S Queue
  -----
  2961983.gpc-sched JobName              rzon              0 Q batch
gpc-f101n084-$
```



# Example 1 (GPC)

```
gpc-f101n084-$ module load intel openmpi
gpc-f101n084-$ mpif90 -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example1
gpc-f101n084-$ cp mycode $SCRATCH/example1
gpc-f101n084-$ cd $SCRATCH/example1
gpc-f101n084-$ cat > myjob.pbs
#!/bin/bash
#PBS -l nodes=8:ppn=8,walltime=1:00:00
#PBS -N JobName
cd $PBS_O_WORKDIR
module load intel openmpi
mpirun -np 64 ./mycode > out
gpc-f101n084-$ qsub myjob.pbs
2961983.gpc-sched
gpc-f101n084-$ qstat (or checkjob 2961983, or showq -u $USER)
  Job id              Name              User Time Use S Queue
  -----
  2961983.gpc-sched JobName              rzon              0 Q batch
gpc-f101n084-$ ls
JobName.e2961983  JobName.o2961983  mycode  myjob.pbs
out
```

# Example 2 (GPC)

gpc-f101n084-\$

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
```

```
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
```

```
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
```

```
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel  
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode  
gpc-f101n084-$ mkdir $SCRATCH/example2  
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$ cd $SCRATCH/example2
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$ cd $SCRATCH/example2
gpc-f101n084-$ cat > joblist.txt
    mkdir run1; cd run1; ../mycode 1 > out
    mkdir run2; cd run2; ../mycode 2 > out
    mkdir run3; cd run3; ../mycode 3 > out
    ...
    mkdir run64; cd run64; ../mycode 64 > out
gpc-f101n084-$
```



# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$ cd $SCRATCH/example2
gpc-f101n084-$ cat > joblist.txt
    mkdir run1; cd run1; ../mycode 1 > out
    mkdir run2; cd run2; ../mycode 2 > out
    mkdir run3; cd run3; ../mycode 3 > out
    ...
    mkdir run64; cd run64; ../mycode 64 > out
gpc-f101n084-$ cat > myjob.pbs
    #!/bin/bash
    #PBS -l nodes=1:ppn=8,walltime=24:00:00
    #PBS -N ASerialJob
    cd $PBS_O_WORKDIR
    module load intel gnu-parallel
    parallel -j 8 < joblist.txt
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$ cd $SCRATCH/example2
gpc-f101n084-$ cat > joblist.txt
    mkdir run1; cd run1; ../mycode 1 > out
    mkdir run2; cd run2; ../mycode 2 > out
    mkdir run3; cd run3; ../mycode 3 > out
    ...
    mkdir run64; cd run64; ../mycode 64 > out
gpc-f101n084-$ cat > myjob.pbs
    #!/bin/bash
    #PBS -l nodes=1:ppn=8,walltime=24:00:00
    #PBS -N ASerialJob
    cd $PBS_O_WORKDIR
    module load intel gnu-parallel
    parallel -j 8 < joblist.txt
gpc-f101n084-$ qsub myjob.pbs
    2961985.gpc-sched
gpc-f101n084-$
```

# Example 2 (GPC)

```
gpc-f101n084-$ module load intel
gpc-f101n084-$ ifort -O3 -xhost mycode.f90 -o mycode
gpc-f101n084-$ mkdir $SCRATCH/example2
gpc-f101n084-$ cp mycode $SCRATCH/example2
gpc-f101n084-$ cd $SCRATCH/example2
gpc-f101n084-$ cat > joblist.txt
    mkdir run1; cd run1; ../mycode 1 > out
    mkdir run2; cd run2; ../mycode 2 > out
    mkdir run3; cd run3; ../mycode 3 > out
    ...
    mkdir run64; cd run64; ../mycode 64 > out
gpc-f101n084-$ cat > myjob.pbs
    #!/bin/bash
    #PBS -l nodes=1:ppn=8,walltime=24:00:00
    #PBS -N ASerialJob
    cd $PBS_O_WORKDIR
    module load intel gnu-parallel
    parallel -j 8 < joblist.txt
gpc-f101n084-$ qsub myjob.pbs
    2961985.gpc-sched
gpc-f101n084-$ ls
    ASerialJob.e2961985  ASerialJob.o2961985  joblist.txt  mycode
    myjob.pbs            run1/                run2/        run3/
```

## Part IV

### DATA MANAGEMENT TIPS

# I/O strategies

- Do not read and write lots of small amounts of data to disk.  
Reading data in from one 4MB file can be enormously faster than from 100 40KB files.
- Write data out in binary. Faster and takes less space.
- Each process writing to a file of its own is not scalable.  
A directory gets locked by the first process accessing it, so the other processes have to wait for it.
- If you must read and write a lot to disk, use ramdisk if possible.  
The ramdisk can be accessed using `/dev/shm/` and is currently set to 11GB max.
- Copy back from ramdisk at end of run.

# Storage Limits at SciNet

location	quota	block-size	time-limit	backup	devel	comp
/home	10GB	256kB	perpetual	yes	rw	ro
/scratch	20TB/ <b>1M</b>	4MB	<b>3 months</b>	<b>no</b>	rw	rw

- Compute nodes do not have local hard drives.
- /home and /scratch are both part of the GPFS file system.
- GPFS is a high-performance file system which provides rapid reads and writes to large data sets in parallel from many nodes.
- Performs poorly accessing data sets which consist of many, small files.
- Avoid reading and writing lots of small amounts of data to disk.
- Many small files on the system would waste space and would be slower to access, read and write.

[http://wiki.scinethpc.ca/wiki/index.php/Data\\_Management](http://wiki.scinethpc.ca/wiki/index.php/Data_Management)

# Moving large data

## *Moving less than 10GB through the login nodes*

- Only login nodes visible from outside SciNet (1Gb/s link).
- Use scp or rsync.
- but datamover1 node is faster.

## *Moving more than 10GB through the datamover1 node*

- Should be done from the datamover1 node (10Gb/s link).
- From any SciNet node, ssh to datamover1.
- Transfers must originate from datamover1.
- Your machine must be reachable from the outside.

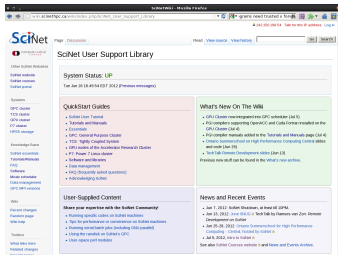
## *Moving data to HPSS*

- HPSS is a tape-based storage solution.
- 1-2TB per group by default (more with special allocation).

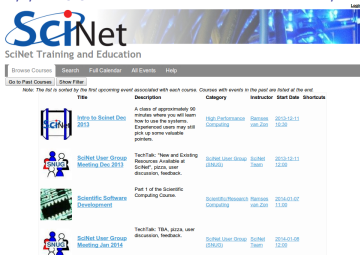
## Part V

### USEFUL SITES





<https://support.scinet.utoronto.ca/education> <https://portal.scinet.utoronto.ca>



SciNet usage reports  
Change password, default allocation, mailist subscriptions





The Summer School on Scientific and High Performance Computing provides attendees with unique opportunities to learn and share knowledge and experience in high performance and technical computing on platforms with the latest technologies. The five full day summer school offers intensive courses on a number of selected subjects...

- » About
- » Programme
- » Registration
- » Location
- » Travel
- » Gallery
- » Past Summer Schools
- » Updates
- » Contact

## 2014 Summer School on High Performance Computing

University of Waterloo, Waterloo, Ontario, May 26-30

University of Toronto, Toronto, Ontario, June 9-13



### Programme

Select your location:

May 26 - 30, [West - SHARCNET](#)

University of Waterloo  
Waterloo, Ontario

June 9 - 13, [Central - SciNet](#)

University of Toronto (St. George Campus)  
Toronto, Ontario

July 7 - 11, East - HPCVL

Carleton University  
Ottawa, Ontario

---

Copyright(C) 2014 Shared Hierarchical Academic Research Computing Network.